

Easy Sets and Hard Certificate Schemes

*Lane A. Hemaspaandra*¹
Department of Computer Science
University of Rochester
Rochester, NY 14627
lane@cs.rochester.edu

*Jörg Rothe*²
Institut für Informatik
Friedrich-Schiller-Universität Jena
07743 Jena, Germany
rothe@informatik.uni-jena.de

*Gerd Wechsung*³
Institut für Informatik
Friedrich-Schiller-Universität Jena
07743 Jena, Germany
wechsung@minet.uni-jena.de

¹Supported in part by grants NSF-INT-9513368/DAAD-315-PRO-fo-ab, NSF-CCR-8957604, NSF-INT-9116781/JSPS-ENGR-207, and NSF-CCR-9322513. Work done in part while visiting the Friedrich-Schiller-Universität Jena.

²Supported in part by grant NSF-INT-9513368/DAAD-315-PRO-fo-ab. Work done in part while visiting the University of Rochester and Le Moyne College, Syracuse.

³Supported in part by grant NSF-INT-9513368/DAAD-315-PRO-fo-ab. Work done in part while visiting the University of Rochester and Le Moyne College, Syracuse.

Abstract

Can easy sets only have easy certificate schemes? In this paper, we study the class of sets that, for all NP certificate schemes (i.e., NP machines), always have easy acceptance certificates (i.e., accepting paths) that can be computed in polynomial time. We also study the class of sets that, for all NP certificate schemes, infinitely often have easy acceptance certificates.

In particular, we provide equivalent characterizations of these classes in terms of relative generalized Kolmogorov complexity, showing that they are robust. We also provide structural conditions—regarding immunity and class collapses—that put upper and lower bounds on the sizes of these two classes. Finally, we provide negative results showing that some of our positive claims are optimal with regard to being relativizable. Our negative results are proven using a novel observation: we show that the classical “wide spacing” oracle construction technique yields instant non-bi-immunity results. Furthermore, we establish a result that improves upon Baker, Gill, and Solovay’s classical result that $\text{NP} \neq \text{P} = \text{NP} \cap \text{coNP}$ holds in some relativized world.

1 Introduction

Borodin and Demers [BD76] proved the following result.

Theorem 1.1 [BD76] If $\text{NP} \cap \text{coNP} \neq \text{P}$, then there exists a set L such that

1. $L \in \text{P}$,
2. $L \subseteq \text{SAT}$, and
3. For no polynomial-time computable function f does it hold that: for each $F \in L$, $f(F)$ outputs a satisfying assignment of F .

That is, under a hypothesis most complexity theoreticians would guess to be true, it follows that there is a set of satisfiable formulas for which it is trivial to determine they are satisfiable, yet it is hard to determine why (i.e., via what satisfying assignment) they are satisfiable.

Motivated by their work, this paper seeks to study, complexity-theoretically, the classes of sets that do or do not have easy certificates. In particular, we are interested in the following four classes. $\text{EASY}_{\forall}^{\forall}$ is the class of sets L such that for each NP machine M accepting them, there is a polynomial-time computable function f_M such that for each $x \in L$, $f_M(x)$ outputs an accepting path of $M(x)$. That is, $\text{EASY}_{\forall}^{\forall}$ is the class of sets that for all certificate schemes, have easy certificates for all elements of the set. We can analogously define $\text{EASY}_{\text{io}}^{\forall}$, $\text{EASY}_{\forall}^{\exists}$, and $\text{EASY}_{\text{io}}^{\exists}$. However, we note that $\text{EASY}_{\forall}^{\exists} = \text{P}$ and $\text{EASY}_{\text{io}}^{\exists}$ equals the class of non-P-immune NP sets. Regarding the two $\text{EASY}_{\dots}^{\forall}$ classes, we provide equivalent characterizations of the classes in terms of

relative generalized Kolmogorov complexity, showing that they are robust. We also provide structural conditions—regarding immunity and class collapses—that put upper and lower bounds on the sizes of these two classes. Finally, we provide negative results showing that some of our positive claims are optimal with regard to being relativizable. Our negative results are proven using a novel observation: we show that the classical “wide spacing” oracle construction technique yields instant non-bi-immunity results. Furthermore, we establish a result that improves upon Baker, Gill, and Solovay’s classical result that $\text{NP} \neq \text{P} = \text{NP} \cap \text{coNP}$ holds in some relativized world [BGS75], and that in addition links their result with the above-stated result of Borodin and Demers.

2 Definitions and Robustness

For the standard notations and the complexity-theoretical concepts used in this paper we refer to some standard text book on computational complexity such as [HU79, BDG, BC93, Pap94]. Fix the alphabet $\Sigma = \{0, 1\}$. Σ^* is the set of all strings over Σ . For each string $u \in \Sigma^*$, $|u|$ denotes the length of u . The empty string is denoted by ϵ . As is standard, the notation $\exists^{\text{io}}x$ (respectively, $\forall^{\text{ae}}x$) means “there exist infinitely many x ” (“for all but finitely many x ”). For each set $L \subseteq \Sigma^*$, $\|L\|$ denotes the cardinality of L and $\overline{L} = \Sigma^* - L$ denotes the complement of L . For any class \mathcal{C} of sets, define $\text{co}\mathcal{C} \stackrel{\text{df}}{=} \{L \mid \overline{L} \in \mathcal{C}\}$. $L^{\leq n}$ ($L^{< n}$) is the set of all strings in L having length n (less than or equal to n). Let Σ^n and $\Sigma^{\leq n}$ be shorthands for $(\Sigma^*)^n$ and $(\Sigma^*)^{\leq n}$, respectively. Let FINITE be the class of all finite sets. To encode a pair of strings, we use a polynomial-time computable, one-one, onto pairing function, $\langle \cdot, \cdot \rangle : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$, that has polynomial-time computable inverses. Denoting the set of non-negative integers by \mathbb{N} and using the standard correspondence between Σ^* and \mathbb{N} , we will view $\langle \cdot, \cdot \rangle$ also as a pairing function mapping $\mathbb{N} \times \mathbb{N}$ onto \mathbb{N} . Let \leq_{lex} denote the standard quasi-lexicographical ordering on Σ^* , that is, for strings x and y , $x \leq_{\text{lex}} y$ if either $x = y$, or $|x| < |y|$, or $(|x| = |y| \text{ and there exists some } z \in \Sigma^* \text{ such that } x = z0u \text{ and } y = z1v)$. $x <_{\text{lex}} y$ indicates that $x \leq_{\text{lex}} y$ but $x \neq y$.

We will abbreviate “polynomial-time deterministic (nondeterministic) Turing machine” by DPM (NPM). For any Turing machine M , $L(M)$ denotes the set of strings accepted by M , and the notation $M(x)$ means “ M on input x .” For any oracle Turing machine M and any oracle set A , $L(M^A)$ denotes the set of strings accepted by M relative to A , and the notation $M^A(x)$ means “ M^A on input x .” For any NPM N and any input x , we assume that all paths of $N(x)$ are suitably encoded by strings over Σ . An NPM N is said to be *normalized* if there exists a polynomial q such that for all n , $q(n) \geq n$ and, on each input of length n , all paths of length $q(n)$ exist in the computation of $N(x)$, and $N(x)$ has only paths of length $q(n)$. Unless otherwise stated, all NPMs considered in this paper are required to be normalized. For any NPM N and any input x , we denote

the set of accepting paths of $N(x)$ by $\text{acc}_N(x)$.

P (respectively, NP) is the class of all sets that are accepted by some DPM (NPM). Let P_∞ denote the class P – FINITE of all infinite P sets. FP denotes the class of all polynomial-time computable functions. For any complexity class \mathcal{C} , a set L is said to be \mathcal{C} -immune if L is infinite and no infinite subset of L is in \mathcal{C} . Let \mathcal{C} -immune denote the class of all \mathcal{C} -immune sets. A set L is said to be \mathcal{C} -bi-immune if both L and \overline{L} are \mathcal{C} -immune. Let \mathcal{C} -bi-immune denote the class of all \mathcal{C} -bi-immune sets. For classes \mathcal{C} and \mathcal{D} of sets, \mathcal{D} is said to be \mathcal{C} -immune (respectively, \mathcal{C} -bi-immune) if $\mathcal{D} \cap (\mathcal{C}\text{-immune}) \neq \emptyset$ (respectively, if $\mathcal{D} \cap (\mathcal{C}\text{-bi-immune}) \neq \emptyset$).

As a notational convention, for any NPM N , we will say “ N has always (respectively, N has infinitely often) easy certificates” to mean that (the encoding of) an accepting path of $N(x)$ can be printed in polynomial time for each string $x \in L(N)$ (respectively, for infinitely many $x \in L(N)$). Similarly, we will say “ N has only (respectively, N has infinitely often) hard certificates” to mean that no FP function is able to output (the encoding of) an accepting path of $N(x)$ for each string $x \in L(N)$ (respectively, for infinitely many $x \in L(N)$). This is more formally stated in Definition 2.1 below that introduces the classes $\text{EASY}_\forall^\forall$, $\text{EASY}_{\text{io}}^\forall$, $\text{EASY}_\forall^\exists$, and $\text{EASY}_{\text{io}}^\exists$ of sets for which all (or some) NP certificate schemes accepting the set have always (or infinitely often) easy certificates.

Definition 2.1 Let $L \subseteq \Sigma^*$ be a set.

1. $L \in \text{EASY}_\forall^\forall$ if and only if
 - (a) $L \in \text{NP}$, and
 - (b) $(\forall N) [(N \text{ is NPM with } L(N) = L) \implies (\exists f_N \in \text{FP}) (\forall x \in L) [f_N(x) \in \text{acc}_N(x)]]$.
2. $L \in \text{EASY}_{\text{io}}^\forall$ if and only if either L is finite, or
 - (a) $L \in \text{NP}$, and
 - (b) $(\forall N) [(N \text{ is NPM with } L(N) = L) \implies (\exists f_N \in \text{FP}) (\exists^{\text{io}} x \in L) [f_N(x) \in \text{acc}_N(x)]]$.
3. $L \in \text{EASY}_\forall^\exists$ if and only if
 - (a) $L \in \text{NP}$, and
 - (b) $(\exists \text{NPM } N) [L(N) = L \wedge (\exists f_N \in \text{FP}) (\forall x \in L) [f_N(x) \in \text{acc}_N(x)]]$.
4. $L \in \text{EASY}_{\text{io}}^\exists$ if and only if either L is finite, or
 - (a) $L \in \text{NP}$, and
 - (b) $(\exists \text{NPM } N) [L(N) = L \wedge (\exists f_N \in \text{FP}) (\exists^{\text{io}} x \in L) [f_N(x) \in \text{acc}_N(x)]]$.

- Remark 2.2** 1. It is easy to see that both $\text{EASY}_{\forall}^{\forall}$ and $\text{EASY}_{\forall}^{\exists}$ contain all finite sets. On the other hand, the $\text{EASY}_{\text{io}}^{\forall}$ classes are *defined* so as to also contain all finite sets; this is just for uniformity and since we feel that it is reasonable to require that the finite sets satisfy any suggested notion of “easy sets.” Note that the $\text{EASY}_{\text{io}}^{\forall}$ classes capture the idea that sets having *correct* NP programs *for all inputs* may have *easy to compute* NP certificates just on *infinitely many* inputs, while possibly having only hard to compute NP certificates on infinitely many other inputs. Let us give some more motivation for these two classes. In particular, we are interested in comparing the “io” notions with the corresponding “ \forall ” notions. While $\text{EASY}_{\text{io}}^{\exists}$ turns out to be a class that is reasonably characterizable in terms of immunity (cf. Theorem 2.3.4 below), $\text{EASY}_{\text{io}}^{\forall}$ plays an important role for intermediate conditions between certain immunity and other statements belonging to the implications that will be proven as the main results of the next section and that are summarized in Figure 2.
2. Note also that we can analogously define $\text{EASY}_{\text{ae}}^{\forall}$ and $\text{EASY}_{\text{ae}}^{\exists}$ by using the quantification “ $\forall^{\text{ae}} x \in L$ ” rather than “ $\forall x \in L$ ” in Parts 1 and 3 of the above definition. However, since the classes $\text{EASY}_{\forall}^{\forall}$ and $\text{EASY}_{\forall}^{\exists}$ (as are most complexity classes) are closed under finite variations, it is clear that $\text{EASY}_{\text{ae}}^{\forall} = \text{EASY}_{\forall}^{\forall}$ and $\text{EASY}_{\text{ae}}^{\exists} = \text{EASY}_{\forall}^{\exists}$. Moreover, we show below that $\text{EASY}_{\forall}^{\exists} = \text{P}$ and that $\text{EASY}_{\text{io}}^{\exists}$ equals the class of all non-P-immune NP sets, and we therefore will not further discuss these two classes in this paper.

Theorem 2.3 1. $\text{FINITE} \subseteq \text{EASY}_{\forall}^{\forall} \subseteq \text{EASY}_{\text{io}}^{\forall} \subseteq \text{EASY}_{\text{io}}^{\exists} \subseteq \text{NP}$.

2. $\text{EASY}_{\forall}^{\forall} \subseteq \text{EASY}_{\forall}^{\exists} \subseteq \text{EASY}_{\text{io}}^{\exists}$.
3. $\text{EASY}_{\forall}^{\exists} = \text{P}$.
4. $\text{EASY}_{\text{io}}^{\exists} = \overline{\text{P-immune}} \cap \text{NP}$.

Proof. (1) & (2) The inclusions immediately follow from Definition 2.1.

(3) The inclusion $\text{P} \subseteq \text{EASY}_{\forall}^{\exists}$ holds by definition. The converse inclusion $\text{EASY}_{\forall}^{\exists} \subseteq \text{P}$ also is clear, since if L is any set in $\text{EASY}_{\forall}^{\exists}$ and this is witnessed by NPM N and FP function f_N , then there exists a DPM M that recognizes L as follows. On input x , M simulates that computation path of $N(x)$ that is printed by $f_N(x)$. If $x \in L$, then $f_N(x) \in \text{acc}_N(x)$, and M accepts x . If $x \notin L$, then $f_N(x)$ cannot be an accepting path of $N(x)$, and thus M rejects x .

(4) Let $L \in \text{EASY}_{\text{io}}^{\exists}$ via NPM N and FP function f_N . Clearly, $L \in \text{NP}$, and if L is not a finite set, then the set

$$\{x \in L \mid f_N(x) = y \text{ and } N(x) \text{ accepts on path } y\}$$

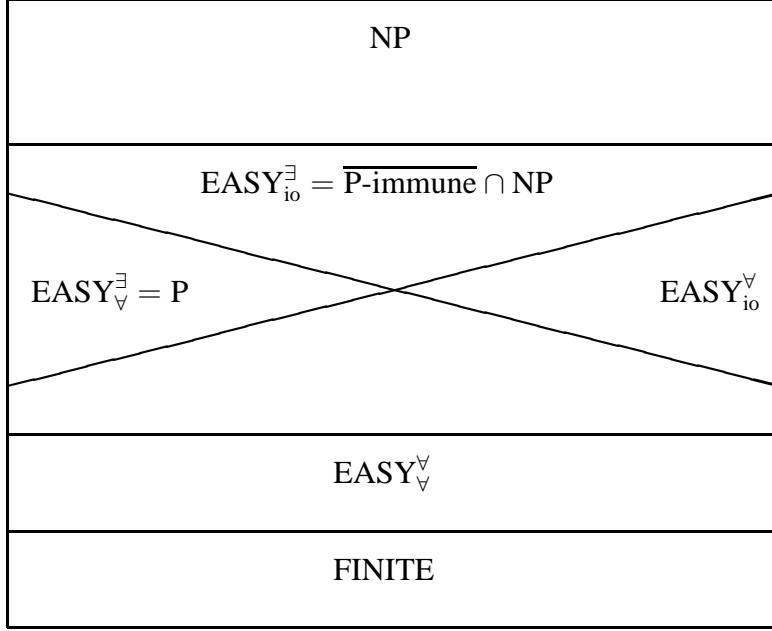


Figure 1: Inclusions between classes of NP sets having easy certificates.

is an infinite subset of L that is in P . Hence, L is not P -immune.

Conversely, let A be any NP set that is not P -immune. Let M_A be an NPM accepting A . If A is finite, then we are done. So suppose A is infinite, and there is an infinite set B such that $B \subseteq A$ and $B \in P$ via DPM M_B . We now describe an NPM N and an FP function f_N that witness $A \in \text{EASY}_{io}^{\exists}$. On input x , N first simulates the computation of $M_B(x)$, and accepts x if M_B accepts x . If M_B rejects x , then N simulates the computation of $M_A(x)$. Clearly, $L(N) = A$. $f_N(x)$ is defined to be the (suitably encoded) computation of $M_B(x)$. Since B is an infinite set, $f_N(x)$ prints an accepting path of $N(x)$ for infinitely many $x \in A$. \square

The inclusion relations between FINITE , NP , and all four classes of easy NP sets ($\text{EASY}_{\forall}^{\forall}$, $\text{EASY}_{\forall}^{\exists}$, $\text{EASY}_{io}^{\forall}$, and $\text{EASY}_{io}^{\exists}$) are displayed in Figure 1.

The Kolmogorov complexity of finite strings was introduced independently by Kolmogorov [Kol65] and Chaitin [Cha66]. Roughly speaking, the Kolmogorov complexity of a finite binary string x is the length of a shortest program that generates x . Intuitively, if a string x can be generated by a program shorter than x itself, then x can be “compressed.” The notion of *generalized* Kolmogorov complexity ([Adl79, Har83, Sip83], see the paper of Li and Vitányi [LV90] for a nice survey of the field) is a version of Kolmogorov complexity that provides information about not only

whether and how far a string can be compressed, but also how fast it can be “restored.” We now give the definition of (unconditional and conditional) generalized Kolmogorov complexity.

Definition 2.4 ([Har83], see also [Adl79, Sip83]) For any Turing machine T and functions s and t mapping \mathbb{N} to \mathbb{N} , define

$$K_T[s(n), t(n)] \stackrel{\text{df}}{=} \{x \mid (\exists y) [|x| = n \text{ and } |y| \leq s(n) \text{ and } T(y) \text{ outputs } x \text{ in at most } t(n) \text{ steps}]\}.$$

It was shown in [Har83] that there exists a *universal*¹ Turing machine U such that for any other Turing machine T there exists a constant c such that

$$K_T[s(n), t(n)] \subseteq K_U[s(n) + c, ct(n) \log t(n) + c].$$

Fixing a universal Turing machine U and dropping the subscript, the *unconditional* generalized Kolmogorov complexity will be denoted by $K[s(n), t(n)] \stackrel{\text{df}}{=} K_U[s(n), t(n)]$. The *conditional* generalized Kolmogorov complexity (under condition z), in which the information of the string z is given for free and does not count for the complexity, is defined as follows.

Definition 2.5 Let U be a fixed universal Turing machine and z be a string. For functions s and t mapping \mathbb{N} to \mathbb{N} , define

$$K[s(n), t(n) \mid z] \stackrel{\text{df}}{=} \{x \mid (\exists y) [|x| = n \text{ and } |y| \leq s(n) \text{ and } U(\langle y, z \rangle) \text{ outputs } x \text{ in } \leq t(n) \text{ steps}]\}.$$

In particular, $K[s(n), t(n) \mid \epsilon] = K[s(n), t(n)]$.

Of particular interest in this paper are certificates (more precisely, strings encoding accepting paths of NPMs) that have *small* generalized Kolmogorov complexity. Levin (see [Tra84]) and Adleman [Adl79] independently discovered the connection between small generalized Kolmogorov complexity and certificates. This connection has also been used in other contexts ([HW91], see also [HR90, GT91] and the comments in [HR90] on [CH89]).

Definition 2.6 [HY84] A set S is *P-printable* if there exists a DPM M such that for each length n , M on input 1^n prints all elements of S having length at most n .

The P-printable sets are closely related to sets of strings having small unconditional generalized Kolmogorov complexity: A set S is P-printable if and only if $S \in \mathbf{P}$ and $S \subseteq K[k \log n, n^k]$ for some constant k ([AR88], see also [BB86, HH88, Rub86]). Below we note a similar connection between

¹Roughly speaking, a universal Turing machine U expects as input a pair of a (suitably encoded) Turing machine T and an input string y and simulates the computation of $T(y)$. More precisely, denoting the encoding of T by $\text{code}(T)$ and using our pairing function, U runs on input $\langle \text{code}(T), y \rangle$ and outputs the result of the computation of $T(y)$.

the sets in $\text{EASY}_{\forall}^{\forall}$ and $\text{EASY}_{\exists}^{\forall}$ and the sets of certificates having small *conditional* generalized Kolmogorov complexity, thus showing the robustness of these notions. Due to Theorem 2.3, the corresponding claims for $\text{EASY}_{\forall}^{\exists}$ and $\text{EASY}_{\exists}^{\exists}$ are omitted. Though the flavor of the correspondence here invoked is by now standard (e.g., see the above papers), we include the proof of Observation 2.7 for completeness.

Observation 2.7 The following are equivalent:

1. $L \in \text{EASY}_{\forall}^{\forall}$.
2. For each normalized NPM N accepting L there is a constant k (which may depend on N) such that for each string $x \in L$ it holds that $\text{acc}_N(x) \cap \mathbf{K}[k \log n, n^k \mid x] \neq \emptyset$.

Proof. In one direction the function proving a machine easy itself yields Kolmogorov-simple certificates. That is, for any normalized NPM N accepting the given $\text{EASY}_{\forall}^{\forall}$ set L , there is an FP function f_N that outputs an accepting path of $N(x)$ for each $x \in L$. Thus, for each $x \in L$, the certificate $f_N(x)$ is in $\mathbf{K}[k \log n, n^k \mid x]$ for some constant k depending only on f_N (and thus on N), since the program for f_N , encoded as a string y , has constant size, and the universal Turing machine U running on input $\langle y, x \rangle$ can clearly generate $f_N(x)$ in time polynomial in $|f_N(x)|$.

In the other direction, let N be any NPM accepting L . By assumption, for each $x \in L$, $N(x)$ has certificates of small conditional Kolmogorov complexity relative to x (i.e., it has certificates in $\mathbf{K}[k \log n, n^k \mid x]$ for some constant k). Note that n , the length of those certificates, is polynomial in $|x|$; let p be some such polynomial bound. So, for each x , $n = p(|x|)$ is a polynomial bound on the length of the certificates of $N(x)$. There are at most $2^{\mathcal{O}(\log n)} = n^{\ell}$ (for some suitable constant ℓ) short strings that potentially encode programs y such that the universal Turing machine U , running on input $\langle y, x \rangle$, produces a certificate of $N(x)$ in time polynomial in n , say in time $n^m = (p(|x|))^m$. Let $q(|x|) \stackrel{\text{df}}{=} \max\{(p(|x|))^{\ell}, (p(|x|))^m\}$.

The function f_N proving N easy works on input x as follows. In a brute-force manner, f_N runs the universal machine on the pairs $\langle y, x \rangle$ for all the at most $q(|x|)$ many short strings y , $|y| \leq k \log n$, for $q(|x|)$ steps, and then for each output checks if the output is an accepting path of $N(x)$, and eventually outputs the first such accepting path found. If no accepting path was found, the input x is not in L . Clearly, f_N is polynomial-time computable and, for each input $x \in L$, outputs a certificate of $N(x)$. \square

Remark 2.8 Note that the normalization requirement in the above observation is crucial, since our definition of conditional generalized Kolmogorov complexity displays the strange feature that for machines that are not normalized, if we use a certain simple polynomial-time computable and

polynomial-time invertible pairing function, say $\langle \cdot, \cdot \rangle_{\text{weird}}$, to encode the pair of the program y and the condition z as input $\langle y, z \rangle_{\text{weird}}$ to the universal Turing machine, then even the empty string has *non-constant* conditional Kolmogorov complexity. Due to our normalization requirement, however, this issue is not germane here.

The proof of Observation 2.9 follows precisely the lines of the proof of Observation 2.7.

Observation 2.9 The following are equivalent:

1. $L \in \text{EASY}_{\text{io}}^{\forall}$.
2. For each normalized NPM N accepting L there is a constant k (which may depend on N) such that for infinitely many strings $x \in L$ it holds that $\text{acc}_N(x) \cap \mathbf{K}[k \log n, n^k \mid x] \neq \emptyset$.

3 Positive Results

In this section, we prove a number of implications between certain properties of subclasses of NP that are summarized in Figure 2. Usually, when one is trying to give strong evidence for some complexity-theoretic statement A not to be true, one does so by showing that A implies $\text{P} = \text{NP}$. In contrast, our Figure 2 has $\text{P} \neq \text{NP}$ as its top conclusion. Nonetheless, the implications of Figure 2 are not meaningless. Their importance is obvious in light of the fact that the statements of the figure (in particular, the immunity assertions and $\text{P} \neq \text{NP} \cap \text{coNP}$ as well as the condition $\text{P} \neq \text{EASY}_{\forall}^{\forall}$, which is equivalent to the existence of surjective one-way functions (see [FFNR96])) are well-studied and important conditions in complexity theory. The implications of Figure 2 simply state that these conditions are at least as hard to prove as proving $\text{P} \neq \text{NP}$, and they explore the logical fine-structure amongst those important conditions.

Here is the key for Figure 2: Implications represented by arrows that are marked by a “*” are not invertible up to the limits of relativizations (as will be shown in Section 4). Consequently, no chain of implications that contains an arrow marked by a “*” is invertible in all relativized worlds. Arrows labeled by boldface numbers indicate non-trivial implications to be proven in Theorem 3.2.

We first discuss the trivial implications of Figure 2. We stress that these trivial statements are included not only in order to make the picture displayed in Figure 2 as complete as possible, but also for the following reason. In the next section, we will prove that the reverse of some implication chains comprising both trivial and more interesting implications (the latter ones being stated in Theorem 3.2 below) not only fails in some relativized worlds, but, even worse, this relativized failure can already be shown for the trivial part of the implication chain considered. Therefore, it does make sense to explicitly state such trivial implications.

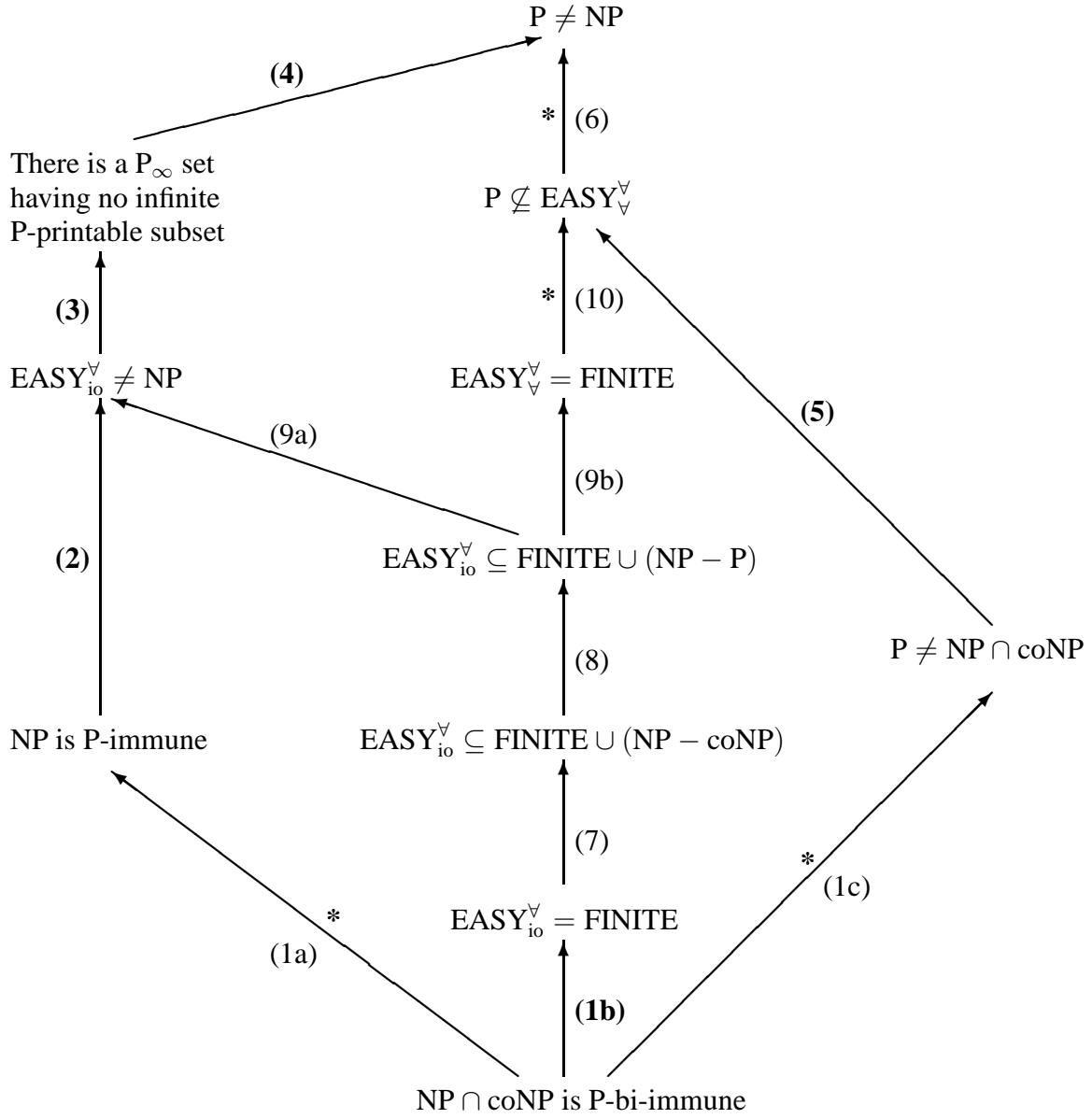


Figure 2: Some implications between various properties of (classes of) sets within NP.

These trivial facts are either immediately clear, or they follow from simple set-theoretical arguments, or are straightforwardly established by the equivalences given in Proposition 3.1 below. For instance, the equivalence of “ $\text{EASY}_{\forall}^{\forall} = \text{FINITE}$ ” and “ $\text{EASY}_{\forall}^{\forall} \subseteq \text{FINITE} \cup (\text{NP} - \text{P})$ ” can be seen by simple set-theoretical considerations.² The statement “ $\text{EASY}_{\forall}^{\forall} = \text{FINITE}$,” in turn, immediately implies the statement “ $\text{P} \not\subseteq \text{EASY}_{\forall}^{\forall}$ ” (see arrow (10) in Figure 2). We have been informed that the authors of [FFNR96] have shown a number of very interesting conditions, including “ $\Sigma^* \notin \text{EASY}_{\forall}^{\forall}$ ” and “there exists an honest polynomial-time computable *onto* function that is not polynomial-time invertible,” to be all equivalent to the statement “ $\text{P} \not\subseteq \text{EASY}_{\forall}^{\forall}$.”

Furthermore, the arrows in Figure 2 labeled (1a), (1c), (7), and (8) are immediately clear. Concerning the arrows (9a) and (9b), note that (9b) follows from the equivalence of “ $\text{EASY}_{\forall}^{\forall} = \text{FINITE}$ ” and “ $\text{EASY}_{\forall}^{\forall} \subseteq \text{FINITE} \cup (\text{NP} - \text{P})$ ” stated in the previous paragraph, whereas (9a) is implied by Proposition 3.1.2 below. Similarly, arrow (6) holds due to Proposition 3.1.1, since if $\text{P} \not\subseteq \text{EASY}_{\forall}^{\forall}$, then there exists a set in $\text{NP} - \text{EASY}_{\forall}^{\forall}$, and thus we have $\text{P} \neq \text{NP}$.

The following proposition gives characterizations for two nodes of Figure 2.

Proposition 3.1 1. $\text{P} \neq \text{NP}$ if and only if $\text{EASY}_{\forall}^{\forall} \neq \text{NP}$.

2. $\text{EASY}_{\text{io}}^{\forall} \subseteq \text{FINITE} \cup (\text{NP} - \text{P})$ if and only if $\Sigma^* \notin \text{EASY}_{\text{io}}^{\forall}$.

Proof. (1) Adleman ([Adl79], see also [Tra84] for a discussion of Levin’s related work) has shown that $\text{P} = \text{NP}$ if and only if for each normalized NPM M there is a k such that for each string $x \in L(M)$ it holds that $\text{acc}_M(x) \cap \text{K}[k \log n, n^k | x] \neq \emptyset$. By Observation 2.7, this implies that $\text{P} = \text{NP}$ if and only if $\text{EASY}_{\forall}^{\forall} = \text{NP}$.

(2) First note that the statement “ $\text{EASY}_{\text{io}}^{\forall} \subseteq \text{FINITE} \cup (\text{NP} - \text{P})$ ” is equivalent to “ $\text{EASY}_{\text{io}}^{\forall} \cap \text{P}_{\infty} = \emptyset$,” and thus immediately implies $\Sigma^* \notin \text{EASY}_{\text{io}}^{\forall}$, since clearly $\Sigma^* \in \text{P}_{\infty}$. For the converse implication, assume there exists a set L in $\text{EASY}_{\text{io}}^{\forall} \cap \text{P}_{\infty}$. Let M_L be a DPM such that $L(M_L) = L$. We show that $\Sigma^* \in \text{EASY}_{\text{io}}^{\forall}$. Let N be any NPM such that $L(N) = \Sigma^*$. By way of contradiction, suppose N has easy certificates only for finitely many $x \in \Sigma^*$. Consider the following NPM N_L for L . On input x , N_L first simulates the computation of $N(x)$, and then, for every path of this simulation, N_L simulates $M_L(x)$ and accepts accordingly. Clearly, $L(N_L) = L$. However, by our supposition that N has easy certificates for finitely many inputs only, N_L also can have easy certificates for at most finitely many inputs, contradicting that $L \in \text{EASY}_{\text{io}}^{\forall}$. Thus, $\Sigma^* \in \text{EASY}_{\text{io}}^{\forall}$. This completes the proof. \square

Next, we prove the non-trivial implications of Figure 2.

²To be definite, for all sets A, B, C , and X , if $A \subseteq X \subseteq B \subseteq C$, then $(X = A \iff X \subseteq A \cup (C - B))$. Taking $A = \text{FINITE}$, $B = \text{P}$, $C = \text{NP}$, and $X = \text{EASY}_{\forall}^{\forall}$, we have verified our claim.

Theorem 3.2 1. If $\text{NP} \cap \text{coNP}$ is P-bi-immune, then $\text{EASY}_{\text{io}}^\forall = \text{FINITE}$.

2. If NP is P-immune, then $\text{EASY}_{\text{io}}^\forall \neq \text{NP}$.

3. If $\text{EASY}_{\text{io}}^\forall \neq \text{NP}$, then there exists an infinite P set having no infinite P-printable subset.

4. [All92] If there exists an infinite P set having no infinite P-printable subset, then $\text{P} \neq \text{NP}$.

5. [BD76] If $\text{NP} \cap \text{coNP} \neq \text{P}$, then $\text{P} \not\subseteq \text{EASY}_{\forall}^\forall$.

Proof. (1) Let Q be any P-bi-immune set such that $Q \in \text{NP} \cap \text{coNP}$ via NPMs N_Q and $N_{\overline{Q}}$, that is, $L(N_Q) = Q$ and $L(N_{\overline{Q}}) = \overline{Q}$. By way of contradiction, assume there exists an infinite set L in $\text{EASY}_{\text{io}}^\forall$. Let N be any NPM accepting L . Consider the following NPM \hat{N} for L . Given x , \hat{N} runs $N(x)$ and rejects on all rejecting paths of $N(x)$. On all accepting paths of $N(x)$, \hat{N} nondeterministically guesses whether $x \in Q$ or $x \in \overline{Q}$, simultaneously guessing certificates (i.e., accepting paths of $N_Q(x)$ or $N_{\overline{Q}}(x)$) for whichever guess was made, and accepts on each accepting path of $N_Q(x)$ or $N_{\overline{Q}}(x)$. Clearly, $L(\hat{N}) = L$. By our assumption that L is an infinite set in $\text{EASY}_{\text{io}}^\forall$, \hat{N} has easy certificates for infinitely many inputs. Let $f_{\hat{N}}$ be an FP function that infinitely often outputs an easy certificate of \hat{N} . Let

$$\hat{L} \stackrel{\text{df}}{=} \{x \mid f_{\hat{N}}(x) \text{ outputs an easy certificate of } \hat{N}(x)\}.$$

Note that \hat{L} is an infinite subset of L , and that for any input x , it can be checked in polynomial time whether x belongs to $Q \cap \hat{L}$ or $\overline{Q} \cap \hat{L}$, respectively, by simply checking whether the string printed by $f_{\hat{N}}$ indeed certifies either $x \in Q \cap \hat{L}$ or $x \in \overline{Q} \cap \hat{L}$. Thus, either $Q \cap \hat{L}$ or $\overline{Q} \cap \hat{L}$ must be an infinite set in P, which contradicts that Q is P-bi-immune. Hence, every set in $\text{EASY}_{\text{io}}^\forall$ is finite.

(2) Let L be any P-immune NP set. We claim that $L \notin \text{EASY}_{\text{io}}^\forall$. Suppose to the contrary that $L \in \text{EASY}_{\text{io}}^\forall$. Let N be any NPM accepting L . Then there exists an FP function f_N such that $f_N(x) \in \text{acc}_N(x)$ for infinitely many inputs x . Define

$$B \stackrel{\text{df}}{=} \{x \mid f_N(x) \in \text{acc}_N(x)\}.$$

Then, B is an infinite subset of L and $B \in \text{P}$, contradicting the P-immunity of L .

(3) If $\text{EASY}_{\text{io}}^\forall \neq \text{NP}$, then there exist an infinite NP set L and an NPM N accepting L such that,

$$(*) \quad (\forall f \in \text{FP}) (\forall^{\text{ae}} x \in L) [f(x) \notin \text{acc}_N(x)].$$

Let q be a polynomial such that $|\langle x, y \rangle| = q(|x|)$ for any string x and any path y of $N(x)$. Define

$$D \stackrel{\text{df}}{=} \{\langle x, y \rangle \mid y \in \text{acc}_N(x)\}.$$

Clearly, D is an infinite set in P . Suppose there exists an infinite set A such that $A \subseteq D$ and A is P -printable via some DPM M . Define an FP function f_A that is computed by DPM M_A as follows. On input x , M_A simulates the computation of $M(1^{q(|x|)})$ and prints all elements of A up to length $q(|x|)$. If a string of the form $\langle x, y \rangle$ is printed, M_A outputs y . Clearly, $f_A(x) \in \text{acc}_N(x)$ for infinitely many $x \in L$, contradicting $(*)$ above. Hence, D has no infinite P -printable subset.

(4) This implication can be seen from results due to Allender [All92]. First, some definitions are needed. Let us consider another version of time-bounded Kolmogorov complexity, a version that is due to Levin [Lev84] (see also [Lev73]). For the fixed universal Turing machine U and any string x , define $\text{Kt}(x)$ to be

$$\min\{|y| + \log n \mid U(y) \text{ outputs } x \text{ in at most } n \text{ steps}\}.$$

For any set L , let $\text{K}_L(n) \stackrel{\text{df}}{=} \min\{\text{Kt}(x) \mid x \in L^{\leq n}\}$. As is standard, let E and NE denote respectively $\bigcup_{c>0} \text{DTIME}(2^{cn})$ and $\bigcup_{c>0} \text{NTIME}(2^{cn})$. An *NE predicate* is a relation R defined by an NE machine M : $R(x, y)$ is true if and only if y encodes an accepting path of $M(x)$. An NE predicate R is *E-solvable* if there is some function f computable in time 2^{cn} for some constant c such that $(\forall x) [(\exists y) [R(x, y)] \iff R(x, f(x))]$.

In [All92], Allender proves that (a) there exists an infinite P set having no infinite P -printable subset if and only if there exists a set $B \in P$ such that $\text{K}_B(n) \in \omega(\log n)$, and (b) there exists an NE predicate that is not E -solvable if and only if there exists a set $C \in P$ such that $\text{K}_C(n) \notin \mathcal{O}(\log n)$. Since $\text{K}_B(n) \in \omega(\log n)$ clearly implies $\text{K}_B(n) \notin \mathcal{O}(\log n)$ and since the existence of an NE predicate that is not E -solvable implies $P \neq NP$, (4) is proven.

For completeness and to enhance readability, we add a more transparent direct proof of (4). To prove the contrapositive, assume $P = NP$. Let L be any infinite set in P . Define the set

$$A \stackrel{\text{df}}{=} \{\langle 0^n, w \rangle \mid n \geq 0 \wedge |w| = n \wedge (\exists z \in L^{\leq n}) [z <_{\text{lex}} w]\}.$$

Clearly, $A \in NP$, and by our assumption, $A \in P$. Define the set of the lexicographically smallest length n strings of L for each length n :

$$S \stackrel{\text{df}}{=} \{x \in L \mid (\forall y \in L^{\leq |x|}) [x \leq_{\text{lex}} y]\}.$$

Clearly, S is an infinite subset of L . Furthermore, S is P -printable, since we can find, at each length n , the lexicographically smallest length n string in L (which is the length n string of S) via prefix search that can be performed in $\text{FP}^A = \text{FP}$. Thus, every infinite set in P has an infinite P -printable subset, as was to be shown.

(5) The proof of this result is implicit in the most common proof (often credited as Hartmanis's simplification of the proof of Borodin and Demers) of the theorem of Borodin and Demers [BD76],

here stated as Theorem 1.1 (see [Sel88] for related work bearing upon the theorem of Borodin and Demers), as has been noted independently of the present paper by Fenner et al. [FFNR96]. For completeness, we include the proof that $\text{NP} \cap \text{coNP} \neq \text{P}$ implies $\text{P} \not\subseteq \text{EASY}_{\forall}^{\forall}$. Let $L \in \text{NP} \cap \text{coNP}$ via NPMs N_L and $N_{\overline{L}}$, that is, $L(N_L) = L$ and $L(N_{\overline{L}}) = \overline{L}$. Assume further that $L \notin \text{P}$. Consider the following NPM M . On input x , M nondeterministically guesses whether $x \in L$ or $x \in \overline{L}$, simultaneously guessing certificates (i.e., accepting paths of $N_L(x)$ or $N_{\overline{L}}(x)$) for whichever guess was made. Now, $L(M) = L(N_L) \cup L(N_{\overline{L}}) = L \cup \overline{L} = \Sigma^*$. Clearly, $\Sigma^* \in \text{P}$. We claim that (under our assumption that $L \notin \text{P}$) $\Sigma^* \notin \text{EASY}_{\forall}^{\forall}$. Suppose to the contrary that $\Sigma^* \in \text{EASY}_{\forall}^{\forall}$. Then, for the NPM M accepting Σ^* , there exists an FP function f_M that prints an accepting path of $M(x)$ on each input x . Hence, L can be decided in polynomial time by simply checking which path of the initial branching of $M(x)$ led to acceptance. That is, a DPM for L , on input x , computes $f_M(x)$ and then checks whether the initial nondeterministic guess of $M(x)$ on the path printed by $f_M(x)$ was either $x \in L$ or $x \in \overline{L}$, and accepts accordingly. This contradicts our assumption that $L \notin \text{P}$. Hence, $\Sigma^* \notin \text{EASY}_{\forall}^{\forall}$. \square

Finally, we state an interesting observation by Selman. Recall that $\text{P} = \text{EASY}_{\forall}^{\forall}$ if and only if $\Sigma^* \in \text{EASY}_{\forall}^{\forall}$. The following claim gives further characterizations of $\text{P} = \text{EASY}_{\forall}^{\forall}$ in terms of the question of whether $\text{EASY}_{\forall}^{\forall}$ is closed under complementation.

Claim 3.3 [Sel95] The following are equivalent.

1. $\text{P} = \text{EASY}_{\forall}^{\forall}$.
2. $\text{EASY}_{\forall}^{\forall}$ is closed under complementation.
3. There exists a set L in P such that $L \in \text{EASY}_{\forall}^{\forall}$ and $\overline{L} \in \text{EASY}_{\forall}^{\forall}$.

Proof. Clearly, Statement (1) implies (2) and (2) implies (3). Assume $L \in \text{P}$, $L \in \text{EASY}_{\forall}^{\forall}$, and $\overline{L} \in \text{EASY}_{\forall}^{\forall}$. Let M_1 (respectively, M_2) be a DPM that accepts L (respectively, \overline{L}). Let N be an NPM that accepts Σ^* . Define NPM N_1 so that on input x , N_1 simultaneously simulates N and M_1 , and N_1 accepts if and only if both N and M_1 accept. Observe that every accepting computation of N_1 encodes an accepting computation of N . Similarly, define N_2 to simultaneously simulate N and M_2 . Then, $L(N_1) = L$ and $L(N_2) = \overline{L}$. Thus, there exist f_1 and f_2 in FP such that $x \in L$ implies $f_1(x)$ is an accepting computation of N_1 , and $x \in \overline{L}$ implies $f_2(x)$ is an accepting computation of N_2 . Define $f(x) = f_1(x)$ if $x \in L$, and $f(x) = f_2(x)$ if $x \in \overline{L}$. Then, $f \in \text{FP}$ and for all x , $f(x)$ contains an encoding of a computation of N on x . Thus, $\Sigma^* \in \text{EASY}_{\forall}^{\forall}$. \square

Consider the reverse of arrow (10) in Figure 2, i.e., the question of whether $\text{P} \not\subseteq \text{EASY}_{\forall}^{\forall}$ implies $\text{EASY}_{\forall}^{\forall} = \text{FINITE}$. Suppose not. That is, suppose that $\text{P} \neq \text{EASY}_{\forall}^{\forall} \neq \text{FINITE}$. Then, there is a

set L such that L is infinite, \overline{L} is infinite, $L \in P$, $L \in \text{EASY}_{\forall}^{\forall}$, and $\overline{L} \notin \text{EASY}_{\forall}^{\forall}$. In Corollary 4.2 below, we will give an oracle relative to which $P \neq \text{EASY}_{\forall}^{\forall} \neq \text{FINITE}$. Since Claim 3.3 and the above comments relativize, in this world, such a set L indeed exists.

4 Negative Results

In this section, we show that some of the results from the previous section are optimal with respect to relativizable techniques. That is, for some of the implications displayed in Figure 2, we construct an oracle relative to which the reverse of that implication does not hold. For instance, from Parts (2) and (5) of Theorem 3.2 and the trivial facts that are shown as arrows (1a) and (1c) in Figure 2, we have the following implication chains:

1. If $\text{NP} \cap \text{coNP}$ is P -bi-immune, then NP is P -immune, which in turn implies that $\text{EASY}_{\text{io}}^{\forall} \neq \text{NP}$,
and
2. If $\text{NP} \cap \text{coNP}$ is P -bi-immune, then $\text{NP} \cap \text{coNP} \neq P$, which in turn implies that $P \not\subseteq \text{EASY}_{\forall}^{\forall}$.

First, we prove that the reverse of these chains fails to hold in some relativized world, and, even worse, that this relativized failure can be shown via proving that not even the trivial parts of the chains are invertible for all oracles. For both chains, this result can be achieved via one and the same oracle to be constructed in the proof of Theorem 4.1 below. This relativized world will additionally satisfy that the inequalities $\text{FINITE} \neq \text{EASY}_{\forall}^{\forall} \neq P \neq \text{NP}$ simultaneously hold in this world (see Corollary 4.2).

One main technical contribution in the proof of Theorem 4.1 is that we give a novel application of the classic “wide spacing” oracle construction technique: We show that this technique *instantly* yields the non- P -bi-immunity of NP relative to some oracle. The use of the wide spacing technique dates so far back that it is hard to know for sure where it was first used. It certainly played an important role in the important early paper by Kurtz [Kur83] (see also the very early use of wide gaps to facilitate the brute-force computation of smaller strings employed by Ladner [Lad75], and also in Baker, Gill, and Solovay’s seminal work [BGS75] and in Rackoff’s oracle constructions for probabilistic and unambiguous polynomial time classes [Rac82]).

Theorem 4.1 There exists a recursive oracle A such that

- (a) $\text{NP}^A = \text{PSPACE}^A$,
- (b) NP^A is P^A -immune, and
- (c) NP^A is not P^A -bi-immune.

Proof. The oracle A will be $\text{QBF} \oplus B$, where QBF is any fixed PSPACE-complete problem and the set B is constructed in stages, $B \stackrel{\text{df}}{=} \bigcup_{j \geq 0} B_j$. Define the function t and the sets T and T_k for $k \geq 0$ by

$$t(0) \stackrel{\text{df}}{=} 2, \quad t(j) \stackrel{\text{df}}{=} 2^{2^{2^{t(j-1)}}} \quad \text{for } j \geq 1, \quad T_k \stackrel{\text{df}}{=} \Sigma^{t(k)} \quad \text{for } k \geq 0, \quad \text{and } T \stackrel{\text{df}}{=} \bigcup_{k \geq 0} T_k.$$

The construction of B will satisfy the following requirement:

$$B \subseteq T \quad \text{and} \quad \|B \cap T_k\| = 1 \quad \text{for each } k \geq 0. \quad (1)$$

Fix an enumeration $\{M_j\}_{j \geq 1}$ of all DPOMs. For each $j \geq 1$, let p_j be a fixed polynomial bounding the runtime of machine M_j . Without loss of generality, assume that our enumeration satisfies for all $j \geq 1$ that

$$\sum_{i=1}^{\lceil \log j \rceil} p_i(0^{t(j)}) < 2^{t(j)-1}.$$

Note that this can indeed be assumed, w.l.o.g., by clocking the machines with appropriately slow clocks as is standard. At stage j of the construction, machines $M_1, M_2, \dots, M_{\lceil \log j \rceil}$ will be *active* unless they have already been canceled during earlier stages. Define the language

$$L_B \stackrel{\text{df}}{=} \{0^n \mid B \cap \Sigma^{n-1}0 \neq \emptyset\}.$$

Clearly, L_B is in NP^B and therefore in NP^A . Let B_{j-1} be the content of B prior to stage j . Initially, let B_0 be the empty set. Stage $j > 0$ of the construction of B is as follows.

Stage j .

Case 1: For no active machine M_i does $M_i^{\text{QBF} \oplus B_{j-1}}(0^{t(j)})$ accept. Choose the smallest string $w_j \in \Sigma^{t(j)-1}0$ such that w_j is not queried in the computation of $M_i^{\text{QBF} \oplus B_{j-1}}(0^{t(j)})$ for any active machine M_i . Set $B_j := B_{j-1} \cup \{w_j\}$.

Case 2: There exists an active machine M_i such that $M_i^{\text{QBF} \oplus B_{j-1}}(0^{t(j)})$ accepts. Let \tilde{i} be the smallest such i . Mark machine $M_{\tilde{i}}$ as canceled, and set $B_j := B_{j-1} \cup \{1^{t(j)}\}$.

End of Stage j .

Since we assume our enumeration of DPOMs to be slowed down so that the sum of the runtimes of all machines that can be active at stage j and run on input $0^{t(j)}$ is strictly less than $2^{t(j)-1}$, the string w_j , if needed, indeed exists. In addition, our assumption on having slowed down the enumeration of machines combined with the widely spaced gaps between the lengths of the strings considered in subsequent stages also guarantees that the single stages of the construction do not interfere with each other, since for each stage j , no machine that is active at this stage can reach

(and query) any string of length $t(j+1)$, that is, no oracle extension at later stages can effect the computations performed in stage j .

Note further that Case 1 in this construction happens infinitely often, as each Case 2 cancels a machine, but at stage j at most $\lceil \log j \rceil$ machines have been active, so Case 2 can happen at most $\lceil \log j \rceil$ times. Since Case 1 happens infinitely often, it is clear that L_B is an infinite set in NP^A . It remains to prove that (a) $\text{NP}^A = \text{PSPACE}^A$, (b) NP^A is P^A -immune, and (c) NP^A is not P^A -bi-immune.

Statement (a) follows immediately from the form of the oracle $A = \text{QBF} \oplus B$ and the fact that QBF is PSPACE-complete.

To prove Statement (b), note that each DPOM M_i is either canceled eventually, or M_i is never canceled. If M_i is canceled, then we have by construction that $0^{t(j)} \in L(M_i^A)$ for some j , yet $0^{t(j)} \notin L_B$, since $B \cap \Sigma^{t(j)-1}0 = \emptyset$. Thus, the language accepted by M_i relative to A , $L(M_i^A)$, is not a subset of L_B . In the other case (i.e., if M_i never is canceled), we will argue that $L(M_i^A) \cap L_B$ must be a finite set. Indeed, let s_i be the first stage in which all machines M_ℓ , with $\ell < i$, that will ever be canceled are already canceled. Then, for no stage j with $j \geq s_i$ will $M_i^{\text{QBF} \oplus B_{j-1}}$ accept the input $0^{t(j)}$, as otherwise M_i would have been the first (i.e., having the smallest number according to the enumeration) active machine accepting $0^{t(j)}$ and would thus have been canceled. It follows that M_i^A accepts at most a finite number (more precisely, at most $s_i - 1$) of the elements of L_B . To summarize, we have shown that there exists an infinite set in NP^A (namely, L_B) having no infinite subset in P^A , that is, NP^A is P^A -immune.

Finally, let us prove Statement (c). Suppose there exists an infinite set L in NP^A . Define the function r by

$$r(0) \stackrel{\text{df}}{=} 2^{2^2} \quad \text{and} \quad r(j) \stackrel{\text{df}}{=} 2^{2^{2^{r(j-1)}}} \quad \text{for } j \geq 1.$$

Define the sets

$$L_{\text{in}} \stackrel{\text{df}}{=} \{0^n \mid (\exists j \geq 0) [r(j) = n \wedge 0^n \in L]\} \quad \text{and} \quad L_{\text{out}} \stackrel{\text{df}}{=} \{0^n \mid (\exists j \geq 0) [r(j) = n \wedge 0^n \notin L]\}.$$

Clearly, $L_{\text{in}} \subseteq L$ and $L_{\text{out}} \subseteq \overline{L}$, and either $0^{r(j)} \in L$ for infinitely many j , or $0^{r(j)} \notin L$ for infinitely many j , or both. Thus, either L_{in} is an infinite subset of L , or L_{out} is an infinite subset of \overline{L} , or both.

Now we prove that both L_{in} and L_{out} are in P^A . Recall that $A = \text{QBF} \oplus B$ and that $B \subseteq T$ and $\|B \cap T_k\| = 1$, since the construction of B satisfies requirement (1) above. We now describe a DPOM M_{in} for which $L(M_{\text{in}}^A) = L_{\text{in}}$. On input x , M_{in}^A first checks whether x is of the form $0^{r(j)}$ for some j . If not, M_{in}^A rejects x . Otherwise, assume $x = 0^{r(k)}$ for some fixed $k \in \mathbb{N}$. Now M_{in}^A constructs a potential query table, Q , of all strings in B that can be touched in the computation of the NP^A machine accepting L . Note that all strings in B that are smaller than $|x| = r(k)$ can

be found by brute force, since—by definition of the functions r and t —they have lengths at least double-exponentially smaller than $|x|$. For the same reason, no string in B of length not smaller than $|x|$ can be touched in the run of the NP^A machine accepting L , on input x , more than finitely often, and all those strings of $B \cap \left(\bigcup_{j \geq k} T_j\right)$ that indeed are queried in this computation can thus be hard-coded into table Q . Therefore, Q contains all information of B that can effect the computation of the NP^A machine for L on input x . Hence, again employing the PSPACE-completeness of QBF, M_{in}^A can ask the QBF part of its oracle to simulate that computation, using table Q for each query to B . If this simulation returns the answer “ $x \in L$,” then M_{in} accepts x ; otherwise, M_{in} rejects x . The proof that $L_{\text{out}} \in \text{P}^A$ is analogous and thus omitted. To summarize, we have shown that if there exists an infinite set L in NP^A , then at least one of L or \bar{L} must contain an infinite subset (specifically, L_{in} or L_{out}) that is decidable in P^A , that is, NP^A is not P^A -bi-immune. \square

Note that the oracle A constructed in the previous proof is recursive and is “reasonable,” since $\text{P} \neq \text{NP}$ holds relative to A , due to the P^A -immunity of NP^A . In addition, relative to A , the reverse of arrows (1a) and (1c) in Figure 2 fails and $\text{FINITE} \neq \text{EASY}_{\forall}^{\forall} \neq \text{P}$, i.e., arrow (10) in Figure 2 is not invertible.

Corollary 4.2 There exists a recursive oracle A such that

1. NP^A is P^A -immune, yet $\text{NP}^A \cap \text{coNP}^A$ is not P^A -bi-immune,
2. $\text{NP}^A \cap \text{coNP}^A \neq \text{P}^A$, yet $\text{NP}^A \cap \text{coNP}^A$ is not P^A -bi-immune,
3. $\text{P}^A \not\subseteq (\text{EASY}_{\forall}^{\forall})^A$, and
4. $(\text{EASY}_{\forall}^{\forall})^A \cap \text{P}_{\infty}^A \neq \emptyset$.

Proof. The first two statements follow from Theorem 4.1, since the non- P^A -bi-immunity of NP^A clearly implies that $\text{NP}^A \cap \text{coNP}^A$ is not P^A -bi-immune. Moreover, since $\text{NP}^A = \text{PSPACE}^A$ implies that $\text{NP}^A = \text{coNP}^A$, there exists a set in $(\text{NP}^A \cap \text{coNP}^A) - \text{P}^A$ by the P^A -immunity of NP^A . Furthermore, since $\text{NP}^A \cap \text{coNP}^A \neq \text{P}^A$, the relativized version of Theorem 3.2.5 establishes the third statement of this corollary: $\text{P}^A \not\subseteq (\text{EASY}_{\forall}^{\forall})^A$.

To prove the fourth statement, let $A = \text{QBF} \oplus B$ be the oracle constructed in the proof of Theorem 4.1. Recall from that proof the definitions of the functions r and t and of the sets T_k , $k \geq 0$, and T . Recall that the construction of B ensures that the following requirement is satisfied: $B \subseteq T$ and $\|B \cap T_k\| = 1$ for every $k \geq 0$. Define the set

$$L \stackrel{\text{df}}{=} \{0^{r(j)} \mid j \geq 0\}.$$

Clearly, L is an infinite set in P , and therefore in P^A .

Let N be any NPOM from our fixed enumeration of all NPOMs (see the proof of Theorem 4.1) that, with oracle $\text{QBF} \oplus B$, accepts L , i.e., $L(N^{\text{QBF} \oplus B}) = L$. To show that N , with oracle $\text{QBF} \oplus B$, has always easy certificates, we will describe a DPOM M that computes, using oracle $\text{QBF} \oplus B$, an $\text{FP}^{\text{QBF} \oplus B}$ function f_N such that f_N prints an accepting path of $N^{\text{QBF} \oplus B}(x)$ for each $x \in L$.

On input x of the form $0^{r(j)}$ for some j , M computes by brute force a potential query table, Q , of all “short” strings in B , i.e., $Q = B^{<r(j)} = B^{\leq t(j)}$, and then employs the PSPACE-completeness of QBF to construct, bit by bit, the lexicographically first accepting path of $N^{\text{QBF} \oplus B}(x)$, say p , via prefix search, where table Q is used to answer all queries to B . Since by definition of r and t , $|x| = r(j)$ is at least double-exponentially smaller than any string of length $> t(j)$, no string in B of length $> t(j)$ can be queried in the run of $N^{\text{QBF} \oplus B}(x)$ more than finitely often, and all those strings in $B \cap \left(\bigcup_{i>t(j)} T_i\right)$ that indeed are queried in this computation can thus be hard-coded into table Q . Therefore, Q contains all information of B that can effect the computation of $N^{\text{QBF} \oplus B}(x)$. It follows that the path p constructed by $M^{\text{QBF} \oplus B}(x)$ indeed is a valid accepting path of $N^{\text{QBF} \oplus B}(x)$. M outputs p . Hence, $L \in (\text{EASY}_{\forall}^{\forall})^A$. This establishes our claim that $(\text{EASY}_{\forall}^{\forall})^A \cap \text{P}_{\infty}^A \neq \emptyset$. \square

Remark 4.3 In fact, it is not hard to see that, via using a Kolmogorov complexity based oracle construction, we can even prove the following claim that is stronger than Corollary 4.2.3 above: There exists an oracle D such that $\text{P}^D \not\subseteq (\text{EASY}_{\text{io}}^{\forall})^D$. To be a bit more precise, in this proof, the set $L \stackrel{\text{df}}{=} \{0^{t(j)} \mid j \geq 0\}$ is clearly a P set (and thus, $L \in \text{P}^X$ for any X), but we can construct an oracle set D such that there exists an NPOM N with $L(N^D) = L$, yet N^D has only hard certificates almost everywhere, i.e., $L \notin (\text{EASY}_{\text{io}}^{\forall})^D$.

Baker, Gill, and Solovay proved that there exists an oracle E relative to which $\text{P}^E \neq \text{NP}^E$, yet $\text{P}^E = \text{NP}^E \cap \text{coNP}^E$ [BGS75]. Due to the priority argument they apply, this proof is the most complicated of all proofs presented in their paper. Next, we show that an adaptation of their proof yields the stronger result that the implication $(\text{P} \neq \text{NP} \implies \text{P} \not\subseteq \text{EASY}_{\forall}^{\forall})$ (which is the reverse of arrow (6) in Figure 2) fails in some relativized world. It is worth noting that thus already the trivial part of the implication chain $(\text{P} \neq \text{NP} \cap \text{coNP} \implies \text{P} \not\subseteq \text{EASY}_{\forall}^{\forall})$ and $(\text{P} \not\subseteq \text{EASY}_{\forall}^{\forall} \implies \text{P} \neq \text{NP})$ (arrows (5) and (6) in Figure 2) is shown to be irreversible up to the limits of relativizing techniques. Furthermore, by inserting into the obvious implication $(\text{P} \neq \text{NP} \cap \text{coNP} \implies \text{P} \neq \text{NP})$ the statement “ $\text{P} \not\subseteq \text{EASY}_{\forall}^{\forall}$,”³ we have, on the one hand, enlightened the close connection between the famous classic results of [BGS75] and [BD76]. On the other hand, having inserted “ $\text{P} \not\subseteq \text{EASY}_{\forall}^{\forall}$ ” into this implication clearly distinguishes the domains where the hard, non-trivial cores of the proofs of the respective results in [BGS75] and [BD76] fall into.

³We stress that it is non-trivial to show that this insertion indeed is possible, due to the commonly agreed non-triviality of Borodin and Demers’s result [BD76] (see Theorem 1.1 and the proof of Theorem 3.2.5).

Theorem 4.4 There exists a recursive oracle A such that $\text{NP}^A \neq \text{P}^A = (\text{EASY}_{\forall}^A)^A$.

Corollary 4.5 [BGS75] There exists a recursive oracle A such that $\text{NP}^A \neq \text{P}^A = \text{NP}^A \cap \text{coNP}^A$.

Proof of Corollary 4.5. This follows from Theorem 4.4 and the fact that the proof of Theorem 3.2.5 relativizes, that is, stating the contrapositive of Theorem 3.2.5: For every oracle B , if $\text{P}^B \subseteq (\text{EASY}_{\forall}^B)^B$, then $\text{P}^B = \text{NP}^B \cap \text{coNP}^B$. \square

Proof of Theorem 4.4. The oracle A will again be $\text{QBF} \oplus B$, where QBF is any fixed PSPACE-complete problem, $B \stackrel{\text{df}}{=} \bigcup_{n \geq 0} B_n$ is constructed in stages, and for every $n > 0$, B_{n-1} denotes the content of B prior to stage n . Initially, set B_0 to be the empty set. To make clear how to construct B , we will recall the crucial parts of Baker, Gill, and Solovay's oracle construction in the proof of [BGS75, Theorem 6], pointing out the differences to our construction.

As in [BGS75], define the function e by $e(0) \stackrel{\text{df}}{=} 2$ and $e(j) \stackrel{\text{df}}{=} 2^{2^{e(j-1)}}$ for $j \geq 1$. At stage n of the construction, at most one string of length $e(n)$ is added to B so as to simultaneously ensure both $\text{P}^A \neq \text{NP}^A$ and $\text{P}^A \subseteq (\text{EASY}_{\forall}^A)^A$.

Fix an enumeration $\{M_j\}_{j \geq 1}$ of all DPOMs and an enumeration $\{N_j\}_{j \geq 1}$ of all NPOMs. For each $j \geq 1$, let p_j be a fixed polynomial bounding the runtime of both M_j and N_j . Define the language

$$L_B \stackrel{\text{df}}{=} \{0^n \mid (\exists w) [w \in B^{=n}]\}.$$

Clearly, L_B is in NP^B , and therefore in NP^A .

There are two types of requirements to be satisfied in the construction of B . Intuitively, satisfying a requirement of the form $\langle i, i \rangle$ will ensure that $L(M_i^A) \neq L_B$. Thus, satisfying $\langle i, i \rangle$ for each $i \geq 1$ will establish our claim that $\text{P}^A \neq \text{NP}^A$. On the other hand, satisfying a requirement of the form $\langle j, k \rangle$ with $j \neq k$ will ensure that $L(N_k^A) \neq L(M_j^A)$, and N_k^A is thus not a machine accepting the P^A set $L(M_j^A)$. Therefore, showing that N_k^A has always easy certificates for every requirement $\langle j, k \rangle$ with $j \neq k$ that is *never* satisfied (and thus the equality $L(N_k^A) = L(M_j^A)$ might happen) will suffice to establish our claim that $\text{P}^A \subseteq (\text{EASY}_{\forall}^A)^A$.

In more detail, an unsatisfied requirement $\langle i, i \rangle$ is *vulnerable* at stage n of the construction of B if $p_i(e(n)) < 2^{e(n)}$. An unsatisfied requirement $\langle j, k \rangle$ with $j \neq k$ is *vulnerable* at stage n if there exists a string x such that

$$e(n-1) < \log |x| \leq e(n) \leq \max\{p_j(|x|), p_k(|x|)\} < e(n+1) \quad (2)$$

and in addition it holds that $x \in L(M_j^{\text{QBF} \oplus B_{n-1}})$ if and only if $x \notin L(N_k^{\text{QBF} \oplus B_{n-1}})$. We note that the definition of vulnerability for this second type of an unsatisfied requirement is different from

that given in the proof of [BGS75, Theorem 6]. By convention, we agree that requirement R_1 has higher priority than requirement R_2 exactly if $R_1 < R_2$. Stage $n > 0$ of the construction of B is as follows.

Stage n . The requirement of highest priority that is vulnerable at stage n will be satisfied. To satisfy requirement $\langle j, k \rangle$ with $j \neq k$, we simply add no string to B in this stage, i.e., $B_n := B_{n-1}$. To satisfy requirement $\langle i, i \rangle$, simulate the computation of $M_i^{\text{QBF} \oplus B_{n-1}}(0^{e(n)})$. If it rejects, then let w_n be the smallest string of length $e(n)$ that is not queried along the computation of $M_i^{\text{QBF} \oplus B_{n-1}}(0^{e(n)})$, and set $B_n := B_{n-1} \cup \{w_n\}$. If $M_i^{\text{QBF} \oplus B_{n-1}}$ accepts $0^{e(n)}$, then set $B_n := B_{n-1}$.

End of Stage n .

Each requirement $\langle i, i \rangle$ is eventually satisfied, since there are only finitely many requirements of higher priority. Suppose requirement $\langle i, i \rangle$ is satisfied at stage n . Then, since $\langle i, i \rangle$ is vulnerable at stage n , we have $p_i(e(n)) < 2^{e(n)}$. This implies that the string w_n , if needed to be added to B in stage n , must exist, and further that no string in B of length $> e(n)$ can be touched in the run of $M_i^A(0^{e(n)})$. Since by construction also w_n is not queried by $M_i^A(0^{e(n)})$, we conclude that oracle extensions at stages $\geq n$ do not effect the computation of $M_i^{\text{QBF} \oplus B_{n-1}}(0^{e(n)})$. Hence, $0^{e(n)} \notin L(M_i^{\text{QBF} \oplus B})$ if and only if $0^{e(n)} \notin L(M_i^{\text{QBF} \oplus B_{n-1}})$ if and only if there exists some string $w_n \in B^{=e(n)}$ if and only if $0^{e(n)} \in L_B$; so $L(M_i^{\text{QBF} \oplus B}) \neq L_B$. It follows that $L_B \in \text{NP}^A - \text{P}^A$.

It remains to prove that $\text{P}^A \subseteq (\text{EASY}_{\forall}^A)^A$. The remainder of this proof is different from the proof in [BGS75]. Given any pair of machines, M_j and N_k with $j \neq k$, we will show that either $L(N_k^A) \neq L(M_j^A)$, or N_k^A has always easy certificates, thus proving that for every set in P^A , each NP^A machine accepting it has always easy certificates, in symbols, $\text{P}^A \subseteq (\text{EASY}_{\forall}^A)^A$.

Clearly, each requirement $\langle j, k \rangle$ with $j \neq k$ is either satisfied eventually, or $\langle j, k \rangle$ is never satisfied. If requirement $\langle j, k \rangle$ is satisfied at stage n for some n , then we are done, since there exists a string x in this case such that (i) $x \in L(M_j^{\text{QBF} \oplus B_{n-1}})$ if and only if $x \notin L(N_k^{\text{QBF} \oplus B_{n-1}})$, (ii) $B_n = B_{n-1}$, and (iii) neither M_j nor N_k can query any string of length $> e(n)$ on input x . Thus, $x \in L(M_j^A)$ if and only if $x \notin L(N_k^A)$, i.e., N_k^A cannot accept the P^A set $L(M_j^A)$. So suppose requirement $\langle j, k \rangle$ is never satisfied, i.e., $L(N_k^A) = L(M_j^A)$ might now happen. Then, it suffices to show that $L(N_k^A) = L(M_j^A)$ implies that N_k^A has always easy certificates. Since this holds for all k for which N_k^A can accept $L(M_j^A)$, we have $L(M_j^A) \in (\text{EASY}_{\forall}^A)^A$.

Let $s_{j,k}$ be the first stage such that

(a) for every x such that $|x| \geq e(s_{j,k})$ there is at most one n such that

$$\log |x| \leq e(n) \leq \max\{p_j(|x|), p_k(|x|)\},$$

(b) all requirements of higher priority than $\langle j, k \rangle$ that will ever be satisfied are already satisfied.

We will now show that N_k^A on input x has easy certificates for every string x accepted by M_j^A . We describe an FP^A function f_k that, on input x , uses oracle $A = \text{QBF} \oplus B$ to output some accepting path of $N_k^A(x)$ if $x \in L(M_j^A)$.

On input x , if $|x| < e(s_{j,k})$, then f_k uses a finite table to find and output some accepting path of $N_k^A(x)$ whenever $x \in L(M_j^A)$. Otherwise (i.e., if $|x| \geq e(s_{j,k})$), f_k calculates the smallest n such that $e(n) \geq \log |x|$. Then, f_k builds a table, T , of all strings that were added to B before stage n , i.e., $T = B^{<e(n)}$, by querying its oracle B about *all* strings of lengths $e(0), e(1), \dots, e(n-1)$. Since $e(n-1) < \log |x|$, only $\mathcal{O}(|x|)$ queries are required in this brute-force search. We have to consider two cases.

Case 1: $e(n) > \max\{p_j(|x|), p_k(|x|)\}$. Then, neither $M_j^A(x)$ nor $N_k^A(x)$ can query their oracle about any string of length $\geq e(n)$. Therefore, the computation of M_j and N_k on input x with oracle $\text{QBF} \oplus T$ is the same as with oracle $\text{QBF} \oplus B$. Hence, f_k can run $M_j^{\text{QBF} \oplus T}$ on input x to determine whether M_j^A would accept x . If it rejects x , then f_k can output an arbitrary string, and we are done. If $M_j^{\text{QBF} \oplus T}$ accepts x , then f_k exploits the PSPACE-completeness of QBF to construct the lexicographically first accepting path of $N_k^{\text{QBF} \oplus B}(x)$, say p , bit by bit via prefix search, where QBF uses table T to answer every oracle call of $N_k^{\text{QBF} \oplus B}(x)$ to B . It follows that p is a valid accepting path of $N_k^A(x)$ if $x \in L(M_j^A)$. f_k outputs p .

Case 2: $e(n) \leq \max\{p_j(|x|), p_k(|x|)\}$. In this case, also strings of length $e(n)$ can be queried by $M_j^A(x)$ or $N_k^A(x)$. Clearly, $N_k^{\text{QBF} \oplus T}$ accepts x if and only if $M_j^{\text{QBF} \oplus T}$ accepts x , as otherwise requirement $\langle j, k \rangle$ would have been satisfied at stage n , contradicting our supposition that $\langle j, k \rangle$ is never satisfied. Indeed, since $\langle j, k \rangle$ is the smallest unsatisfied requirement at stage n by (b) above and since x meets condition (2) above by (a), the equivalence

$$(x \in L(M_j^{\text{QBF} \oplus T}) \iff x \notin L(N_k^{\text{QBF} \oplus T}))$$

would enforce the vulnerability of $\langle j, k \rangle$ at this stage. Now, f_k simulates $M_j^A(x)$ and outputs an arbitrary string if it rejects. Otherwise (i.e., if $x \in L(M_j^A)$), f_k runs $M_j^{\text{QBF} \oplus T}(x)$, call this computation q . There are two subcases.

Case 2.1: The computation of $M_j^A(x)$ exactly agrees with q . Then, there exists an accepting path of $N_k^{\text{QBF} \oplus T}(x)$, and f_k again employs QBF to construct the lexicographically first accepting path of $N_k^{\text{QBF} \oplus T}(x)$, call this path p . If p were reliable w.r.t. oracle A , then f_k could simply output p , and we were done. However, p is not reliable, since T and B might differ, so f_k has to check the validity of p . By our choice of $s_{j,k}$, there exists (according to (a) above) at most one n such that $\log |x| \leq e(n) \leq \max\{p_j(|x|), p_k(|x|)\}$.

Hence, T can lack at most one length $e(n)$ string of B that might be queried in the run of $N_k^A(x)$. Now, f_k checks whether p is a valid certificate of $N_k^A(x)$ by simply checking whether all answers given along p are correct according to the B part of f_k 's oracle. There are two subcases of Case 2.1.

Case 2.1.1: All strings z queried along p receive the answer “yes” if and only if $z \in B$.

We conclude that p is a valid certificate of $N_k^A(x)$. f_k outputs p .

Case 2.1.2: There exists a string z that is queried along p , but receives a wrong “no” answer according to B , i.e., $z \in B$. Then, f_k has detected that z is the one string of length $e(n)$ in $B - T$. So, adding z to T , we now have $T = B^{\leq e(n)}$. f_k again employs QBF to construct the lexicographically first accepting path of $N_k^{\text{QBF} \oplus T}(x)$, say p' , which now must be a valid certificate of $N_k^A(x)$. f_k outputs p' .

Case 2.2: The computation of $M_j^A(x)$ differs from q . The only way this could happen, however, is that the one missing string in T , $z \in B^{\leq e(n)} - T$, is queried on q , but has received a wrong “no” answer from T . Then, as in Case 2.1.2, f_k has identified z and can complete table T by adding z to T . Now, $T = B^{\leq e(n)}$, and f_k can proceed as in Case 2.1.2 to find and output a valid certificate of $N_k^A(x)$ if x is accepted by M_j^A (via once more employing QBF in the prefix search to construct the lexicographically first certificate).

Since $\max\{p_j(|x|), p_k(|x|)\} < e(n+1)$ by (a) above, no string of length $\geq e(n+1)$ can be queried by N_k or M_j on input x , and thus oracle extensions at stages $\geq n+1$ cannot effect the computation of $N_k^{\text{QBF} \oplus B_n}(x)$ or $M_j^{\text{QBF} \oplus B_n}(x)$. This completes the proof. \square

- Remark 4.6** 1. The argument given in the above proof could almost seem to even prove that there exists some oracle A relative to which $P^A \neq \text{NP}^A$ and $\text{NP}^A \subseteq (\text{EASY}_{\forall}^A)^A$, i.e., $P^A \neq \text{NP}^A$ and $P^A = \text{NP}^A$, which clearly is impossible. In fact, our argument would *not* work for NP in place of P , for the following subtle reason. When we define the vulnerability of requirements of the form $\langle j, k \rangle$ with $j \neq k$ in terms of pairs of two *nondeterministic* oracle machines N_j and N_k , and modify our argument appropriately, then the FP^A function f_k has no way of telling whether or not the input x is accepted by N_j^A (as we have no P^A algorithm as in the above proof) and therefore is in serious trouble when it is trying to construct a valid certificate of $N_k^A(x)$.
2. Fortnow and Rogers [FR94] have presented an oracle A such that the reverse of our arrow (5) in Figure 2 fails relative to A , and in fact they show that $P \subseteq \text{EASY}_{\forall}^A$ holds relative to any “sparse generic oracle with the *subset* property.” In fact, regarding their oracle A , Fenner et

al. [FFNR96] note that since they even prove that an intermediate condition (of our arrow (5)) cannot imply $P \neq NP \cap \text{coNP}$, they have a statement slightly stronger than that the reverse of arrow (5) fails relative to A .

Of course, the existence of relativized worlds A in which a statement X^A fails should not be viewed as evidence that X fails in the unrelativized world. Rather, the existence of such relativized worlds should be viewed as evidence that most standard proof techniques lack the power to prove that X holds in the unrelativized world (see, e.g., [All90, For94, Har85] for discussions of how to interpret relativized results). We suggest as an open question the issue of whether even stronger implications than those of Figure 2 can be established.

As a final remark, an anonymous referee mentioned the interesting open topic of definitions analogous to ours, except with the path-finding operator being probabilistic (either error-bounded or error-unbounded), rather than deterministic.

Acknowledgments

We thank Eric Allender for pointing out that if $NP \cap \text{coNP}$ is P-bi-immune, then $\text{EASY}_{\forall}^{\forall} = \text{FINITE}$, for pointing out an alternative (and in fact stronger) proof of Theorem 3.2.4, and for generally inspiring this line of research. We acknowledge interesting discussions with Alan Selman and Lance Fortnow on this subject. In particular, we are indebted to Alan Selman for generously permitting us to include his proof of Claim 3.3 and to Lance Fortnow for providing us with an advance copy of the manuscript [FFNR96].

References

- [Adl79] L. Adleman. Time, space, and randomness. Technical Report MIT/LCS/TM-131, MIT, Cambridge, MA, April 1979.
- [All90] E. Allender. Oracles versus proof techniques that do not relativize. In *Proceedings of the 1990 SIGAL International Symposium on Algorithms*, pages 39–52. Springer-Verlag Lecture Notes in Computer Science #450, August 1990.
- [All92] E. Allender. Applications of time-bounded Kolmogorov complexity in complexity theory. In O. Watanabe, editor, *Kolmogorov Complexity and Computational Complexity*, EATCS Monographs on Theoretical Computer Science, pages 4–22. Springer-Verlag, 1992.
- [AR88] E. Allender and R. Rubinfeld. P-printable sets. *SIAM Journal on Computing*, 17:1193–1202, 1988.
- [BB86] J. Balcázar and R. Book. Sets with small generalized Kolmogorov complexity. *Acta Informatica*, 23(6):679–688, 1986.

- [BC93] D. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity*. Prentice Hall, 1993.
- [BD76] A. Borodin and A. Demers. Some comments on functional self-reducibility and the NP hierarchy. Technical Report TR 76-284, Cornell Department of Computer Science, Ithaca, NY, July 1976.
- [BDG] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity*. Springer-Verlag, Vol. 1 (1988), Vol. 2 (1990).
- [BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the $P=?NP$ question. *SIAM Journal on Computing*, 4:431–442, 1975.
- [CH89] J. Cai and L. Hemachandra. Enumerative counting is hard. *Information and Computation*, 82(1):34–44, 1989.
- [Cha66] G. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13:547–569, 1966.
- [FFNR96] S. Fenner, L. Fortnow, A. Naik, and J. Rogers. On inverting onto functions. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity*, pages 213–222. IEEE Computer Society Press, May 1996.
- [For94] L. Fortnow. The role of relativization in complexity theory. *Bulletin of the EATCS*, (52):229–244, 1994.
- [FR94] L. Fortnow and J. Rogers. Separability and one-way functions. In *Proceedings of the 5th International Symposium on Algorithms and Computation*, pages 396–404. Springer-Verlag *Lecture Notes in Computer Science* #834, August 1994.
- [GT91] F. Green and J. Torán. Kolmogorov complexity of $\#P$ functions. Manuscript, 1991.
- [Har83] J. Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 439–445. IEEE Computer Society Press, 1983.
- [Har85] J. Hartmanis. Solvable problems with conflicting relativizations. *Bulletin of the EATCS*, 27:40–49, 1985.
- [HH88] J. Hartmanis and L. Hemachandra. On sparse oracles separating feasible complexity classes. *Information Processing Letters*, 28:291–295, 1988.
- [HR90] L. Hemachandra and S. Rudich. On the complexity of ranking. *Journal of Computer and System Sciences*, 41(2):251–271, 1990.
- [HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

- [HW91] L. Hemachandra and G. Wechsung. Kolmogorov characterizations of complexity classes. *Theoretical Computer Science*, 83:313–322, 1991.
- [HY84] J. Hartmanis and Y. Yesha. Computation times of NP sets of different densities. *Theoretical Computer Science*, 34:17–32, 1984.
- [Kol65] A. Kolmogorov. Three approaches for defining the concept of information quantity. *Problems Information Transmission*, 1:1–7, 1965.
- [Kur83] S. Kurtz. A relativized failure of the Berman-Hartmanis conjecture. Technical Report TR83-001, University of Chicago Department of Computer Science, Chicago, IL, 1983.
- [Lad75] R. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22(1):155–171, 1975.
- [Lev73] L. Levin. Universal sorting problems. *Problems of Information Transmission*, 9:265–266, 1973.
- [Lev84] L. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61:15–37, 1984.
- [LV90] M. Li and P. Vitányi. Applications of Kolmogorov complexity in the theory of computation. In A. Selman, ed., *Complexity Theory Retrospective*, pages 147–203. Springer-Verlag, 1990.
- [Pap94] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
- [Rac82] C. Rackoff. Relativized questions involving probabilistic algorithms. *Journal of the ACM*, 29(1):261–268, 1982.
- [Rub86] R. Rubinstein. A note on sets with small generalized Kolmogorov complexity. Technical Report TR86-4, Iowa State University, Ames, IA, March 1986.
- [Sel88] A. Selman. Natural self-reducible sets. *SIAM Journal on Computing*, 17(5):989–996, 1988.
- [Sel95] A. Selman. Personal Communication, May, 1995.
- [Sip83] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on Theory of Computing*, pages 330–335, 1983.
- [Tra84] B. Trakhtenbrot. A survey of Russian Approaches to *perebor* (brute-force search) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.